XV

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/464,636	12/15/1999	RICHARD DIEVENDORFF	3382-49606	7885

7590 02/23/2004

KLARQUIST SPARKMAN CAMPBELL
LEIGH & WHINSTON LLP
ONE WORLD TRADE CENTER SUITE1600
121 S W SALMON STREET
PORTLAND, OR 97204

EXAMINER	
ZHEN, LI B	
ART UNIT	PAPER NUMBER
2126	19

DATE MAILED: 02/23/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	**Application No.** 09/464,636	**Applicant(s)** DIEVENDORFF ET AL.
	Examiner Li B. Zhen	**Art Unit** 2126

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _10 November 2003_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-18_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) _13_ is/are allowed.

6)☒ Claim(s) _1-12 and 14-18_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☒ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _17_.	6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1 – 18 are pending in this application.

2. Claim 13 is allowed.

Claim Rejections - 35 USC § 101

3. Claims 6 – 11 are rejected under 35 U.S.C. 101 because they are directed to non-statutory subject matter.

4. Claims 6 – 11 are directed to method steps which can be practiced mentally in conjunction with pen and paper, therefore they are directed to non-statutory subject matter. Specifically, as claimed, it is uncertain what performs each of the claimed method steps. Moreover, each of the claimed steps, inter alia, converting, storing, transferring, constructing, transmitting, can be practiced mentally in conjunctions with pen and paper. The claimed steps do not define a machine or computer implemented process [see MPEP 2106]. Therefore, the claimed invention is directed to non-statutory subject matter. (The examiner suggests applicant to change "method" to "computer implemented methods" in the preamble to overcome the outstanding 35 U.S.C. 101 rejection).

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1 – 12 and 14 – 18 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,567,861 to Kasichainula in view of U.S. Patent No. 6,651,109 to Beck.

7. As to claim 1, Kasichainula teaches the invention substantially as claimed including an object execution system supporting passing of an object reference in a method invocation delivered via a message [a system, method and program product for executing or running objects remotely, some of which objects may pass data streams between themselves; col. 3, lines 1 – 18], the system comprising:

an object configuration store containing object properties information representing properties of at least first and second object classes executable in the system [information in the bytecode file for a class; col. 4, lines 43 – 67], the second object class having a method with a parameter for passing an object reference [pass complex objects as parameters; col. 3, lines 56 – 67];

a method invocation recording facility operative responsive to request of a client program to supply method invocations recorders [PDH and COPH, together with Automatic Object Distribution [AOD], generate the distribution code; col. 4, lines 1 – 15];

a first method invocations recorder [object Z 504, Fig. 5B; col. 7, lines 37 – 50] supplied by the method invocation recording facility at request of the client program [client object X 502, Fig. 5B; col. 7, lines 38 – 49]; and

a second method invocations recorder [proxy Y' 510, Fig. 5B; col. 7, lines 38 – 67] supplied by the method invocation recording facility at request of the client program, the second method invocations recorder operating in response to a method invocation in which an object reference to the first method invocations recorder is passed [Object X 502 is shown making a remote method call to object Y 503, and object Z 504 is one of the parameters; col. 7, lines 55 – 67] to cause a data stream representation of the first method invocations recorder to be marshaled into a method invocations message [object Y' passes the call to object Y via Remote Method Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call; col. 8, lines 19 – 32];

a method invocation play-back facility operative to supply method invocation players [proxy Y" 511, Fig. 5B; col. 8, lines 30 – 50];

a first method invocation player in response to the method invocations message [upon receiving the remote call] to unmarshal the date stream representation [translates the call into the semantics of object Y 503; col. 8, lines 32 – 45] and create therefrom a copy of the first method invocations [creates proxy Z' 512, Fig. 5B] recorder [Upon receiving the remote call from Y' 510, Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509; col. 8, lines 32 – 54], and to pass an object reference to the copy of the first method invocations recorder as a parameter [reference to proxy object Z' 512 is passed 516 as the parameter] of a method invocation to an object of the second object class [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter; col. 8, lines 32 – 52];

wherein an object reference to the copy of the first method invocation recorder is passed as a parameter of the method invocation to an object of the second object class [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter; col. 8, lines 32 – 52] so an object of the second object class can record results for an object of the first object class [Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503; col. 8, line 62 – col. 9, line 5].

8. Although Kasichainula teaches the invention substantially as claimed, Kasichainula does not teach queued method invocations.

However, Beck teaches queued method invocation [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23] between a first object and a second object [col. 1, lines 60 – 67].

9. It would have been obvious to a person of ordinarily skilled in the art at the time of the invention to apply the teaching of queued method invocations as taught by Beck to the invention of Kasichainula because queued method invocations allows asynchronous remote procedure calls which permits applications to send a remote method request and continue with other work without waiting for the request to complete.

10. As to claim 9, Kasichainula as modified by Beck teaches yielding results from processing work of a queued component to a persist-able object [Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503; col. 8, line 62 – col. 9, line 5 of Kasichainula], where the work of the queued component is initiated by method

invocations delivered via a first message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck], the method comprising:

responsive to a client program issuing a set of method invocations for the queued component, marshaling data for the method invocations of the set into a message [call 505 is actually made to proxy object Y' 510...passing the call via Remote Method Invocation or some other standard remote calling method to Y" 511; col. 7, lines 55 – 67 of Kasichainula]; and

when marshaling a reference for calling methods on the persist-able object in any of the method invocations issued by the client program for the queued component [Object X 502 is shown making a remote method call to object Y 503, and object Z 504 is one of the parameters; col. 7, lines 55 – 67 of Kasichainula], persisting the persist-able object into an object-representative data stream and incorporating the object-representative data stream in the data marshaled into the message [object Y' passes the call to object Y via Remote Method Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call; col. 8, lines 19 – 32 of Kasichainula];

submitting the message to the first message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck]; and

at a later time of processing the message from the first message queue [upon receiving the remote call], unmarshaling the data for the method invocations from the message [translates the call into the semantics of object Y 503; col. 8, lines 32 – 45 of Kasichainula], re-creating the persist-able object [creates proxy Z' 512, Fig. 5B of Kasichainula] from the object representative data stream [Upon receiving the remote call from Y' 510, Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509; col. 8, lines 32 – 54 of Kasichainula], issuing the set of method invocations to the queued component [invoke object Y; col. 8, lines 32 – 52 of Kasichainula], and passing a reference for calling methods on the re-created persist-able object to the queued component [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter; col. 8, lines 32 – 52 of Kasichainula].

11. As to claim 16, Kasichainula as modified by Beck teaches a queued method invocations playing component [proxy Y" 511, Fig. 5B; col. 8, lines 30 – 50 of Kasichainula] operating to retrieve a method invocations message from a message queue associated with a first queued component [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck], the first queued component having a reference passing method accepting a passed object reference as a parameter [object Y' passes the call to object Y via Remote Method Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call; col. 8, lines 19 – 32 of Kasichainula], the queued method invocations playing component further operating in response to a message containing a data stream representative of an invocation of the reference passing method [Upon receiving the remote call from Y' 510;

col. 8, lines 32 – 54 of Kasichainula] having a reference for a method invocation recording component of a second queued component passed as the parameter [a reference to Z" is provided in place of Z as the parameter in the call; col. 8, lines 19 – 32 of Kasichainula] to unmarshal the data stream [translates the call into the semantics of object Y 503; col. 8, lines 32 – 45 of Kasichainula], to re-create the method invocation recording component [Upon receiving the remote call from Y' 510, Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509; col. 8, lines 32 – 54 of Kasichainula], and to invoke the method on the first queued component with a reference for the recreated method invocation recording component passed as the parameter [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter; col. 8, lines 32 – 52 of Kasichainula];

wherein the reference to the re-created method invocation recording component is passed to the first queued component [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter; col. 8, lines 32 – 52 of Kasichainula] so the first queued component can record a result to the second queued component [Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503; col. 8, line 62 – col. 9, line 5 of Kasichainula].

12. As to claim 14, Kasichainula as modified by Beck teaches a queued component [an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck] recorder constructor operating on request of a client program to obtain a queued

component reference to create a method invocation recording component [PDH and COPH, together with Automatic Object Distribution [AOD], generate the distribution code; col. 4, lines 1 – 15 of Kasichainula];

a first method invocation recording component [object Z 504, Fig. 5B; col. 7, lines 37 – 50 of Kasichainula] created by the queued component recorder constructor responsive to a first request of a client program [client object X 502, Fig. 5B; col. 7, lines 38 – 49 of Kasichainula]; and

a second method invocation recording component [proxy Y' 510, Fig. 5B; col. 7, lines 38 – 67 of Kasichainula] created by the queued component recorder constructor responsive to a second request of the client program to obtain a second reference for a second queued component, the second queued component having a reference passing method accepting a passed object [object Z 504; col. 7, lines 55 – 67 of Kasichainula] reference as a parameter thereto [Object X 502 is shown making a remote method call to object Y 503, and object Z 504 is one of the parameters; col. 7, lines 55 – 67 of Kasichainula], the second method invocation recording component operating in response to an invocation of the reference passing method made on the second method invocation recording component having a reference for the first method invocation recording component [object Z 504; col. 7, lines 55 – 67 of Kasichainula] passed as the parameter [Object X 502 is shown making a remote method call to object Y 503, and object Z 504 is one of the parameters; col. 7, lines 55 – 67 of Kasichainula] to marshal the first method invocation recording component into a data stream representative of the invocation into a message [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter; col. 8, lines 32 – 52] for queuing into a message queue associated with the second queued component [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck].

13. As to claim 6, Kasichainula as modified by Beck teaches a method of yielding results from processing work of a first queued component to a second queued component [Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503; col. 8, line 62 – col. 9, line 5 of Kasichainula], where the work of the first queued component is initiated by method invocations delivered via a first message queue, and the second queued component is dispatched method invocations delivered into a second message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck], the method comprising:

responsive to a client program issuing a first set of method invocations for the first queued component [Object X 502 is shown making a remote method call to object Y 503, and object Z 504 is one of the parameters; col. 7, lines 55 – 67 of Kasichainula], marshaling data for the method invocations of the first set into a message to be enqueued into the first message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck]; and when marshaling an interface pointer reference to the second queued component in any of the method invocations issued by the client program for the first queued component [object Y' examines the call and realizes that complex object Z 504 is one of the parameters; col. 7, lines 55 – 67 of Kasichainula], incorporating interface passing information in the data marshaled into the message [object Y' passes the call to object Y via Remote Method Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call; col. 8, lines 19 – 32 of Kasichainula], the interface passing information [a reference to Z" is provided in place of Z as the parameter in the call; col. 8, lines 19 – 32 of Kasichainula] designating to enqueue any method invocation by the first queued component on an interface of the second queued component referenced by the interface pointer reference into the second message queue [Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503; col. 8, line 62 – col. 9, line 5 of Kasichainula].

14. As to claim 12, Kasichainula as modified by Beck teaches a distributed computing system [distributed objects; column 3, lines 57 – 67; column 6, lines 24 – 50 of Kasichainula], a multiplicity of client machines [client system 401, Fig. 4B of Kasichainula], a server machine [server system 402, Fig. 4B of Kasichainula], a server-side component [server object Y 404, Fig. 4B of Kasichainula], and a client-specific component [client object X 403, Fig. 4B of Kasichainula];

associating a first message queue with the server-side queued component and a second message queue with a client-specific queued component of the client-specific component-based program [an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck];

a client program [object X 502] issuing a first method invocation to the server-side component [object Y 503] having passed therein a reference [object Z 504 is one of the parameters] for a client-specific component [Object X 502 makes a remote method call to object Y 503, and object Z 504 is one of the parameters, Fig. 5B; column 7, lines 55 - 67 of Kasichainula], recording data representative of the first method invocation into a first method invocations message [object Y' 510 creates a proxy object Z" 513 for object Z 504 before passing the call to Y" 511], automatically and transparently marshaling a reference to the second message queue [reference to proxy object Z"] with the data representative of the first method invocation into the first method invocations message [object Y' 510 creates a proxy object Z" 513 for object Z 504 before passing the call to Y" 511...this proxy contains the code necessary to allow a reference to itself to be passed over the network...object Y' passes the call to object Y" via Remote Method Invocation or some other standard remote calling method 514, and a reference to Z" is provided in place of Z as the parameter in the call, Fig. 5B; column 8, lines 19 - 33 of Kasichainula];

submitting the first method invocations message to the first message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck];

retrieving the first method invocations message from the first message queue at the server machine [upon receiving the remote call from Y' 510; column 8, lines 33 - 42 of Kasichainula];

unmarshaling the data representative of the first method invocation from the first method invocations message [Y" 511 creates a proxy Z' 512 for object Z 504 in machine 509, and creates 515 a reference table entry in which the key Z' returns a remote call reference to the proxy Z" 513 which was created by Y' 510...then when object Y" 511 translates the call into the semantics of object Y 503, Fig. 5B; column 33 – 42 of Kasichainula];

invoking per the first method invocation a method of the server-side queued component [invokes object Y], wherein said invoking comprises passing a reference for the client-specific queued component [reference to proxy object Z' 512 is passed as the parameter] to the server-side queued component [invokes object Y, a reference to proxy object Z' 512 is passed 516 as the parameter...thus object Y will invoke object Z' 512 when object Y's invoked method invokes the object passed in as a parameter; column 8, lines 33 - 67 of Kasichainula]; and

on invoking by the server-side queued component a method of the client-specific queued component using the reference passed to the server side queued component [when object Z' is invoked 506 by object Y 503, object Z' uses the reference table entry which was created earlier 515 by object Y" 511 to determine where the call is to be directed; column 8, lines 33 – 67 of Kasichainula], automatically and transparently to the server-side queued component recording data representative of the server-side queued component's method invocations using the reference passed to the server side queued component into a second method invocations message [object Z' translates the call into the semantics of object Z" and invokes object Z" using Remote Method Invocation or some other standard remote calling method 518; column 8, lines 33 – 67 of Kasichainula], and submitting the second method invocations message to the second message queue, whereby the server-side queued component's method invocations are queued for the client-specific queued component [object Y finishes the method which was invoked from object X 502, it returns the result 523, if any, of said invocation to object Y" 511, which returns said result to object Y' 510, which returns 508 said result to object X 502, thus completing the object X's method invocation to object Y 503, which also updated object Z 504; column 8, lines 33 – 67 of Kasichainula].

15. As to claim 2, Kasichainula as modified teaches the first method invocations recorder is marshaled into the data stream representation via a marshal-by-value operation [If a data stream is passed as a parameter to these calls, the present invention enhances the proxies to do the PDH necessary to obtain data from the original data stream object; col. 4, lines 13 – 35 of Kasichainula], such that the copy of the first method invocations recorder can be created on a separate computing machine [send the data that the data stream object contains over the network, and reassemble it in the destination machine into a proxy data stream object that can be accessed as if it were local; col. 4, lines 13 – 35 of Kasichainula].

16. As to claims 3, 10 and 11, Kasichainula as modified teaches a persistence interface associated with the first method invocations recorder, and the second method invocations recorder invoking the persistence interface to cause marshaling of the first method invocations recorder into the data stream representation [object Y' 510 creates a proxy object Z" 513 for object Z 504 before passing the call to Y" 511...proxy contains the code necessary to allow a reference to itself to be passed over the network (for example, in the Java environment it implements the serializable interface); col. 8, lines 19 – 33 of Kasichainula].

17. As to claim 4, Kasichainula as modified teaches the second method invocations recorder further operates to cause an identification of a message queue associated with the first object class to be marshaled into the data stream representation [first direct method call contains information regarding a location in memory of the intermediary object; col. 1, line 60 – col. 2, line 10].

18. As to claim 5, Kasichainula as modified teaches execution of distributed objects across remote machines in a distributed computing system [Automatic Object Distribution (AOD) process is used to distribute the objects and their proxies into the network; col. 6, lines 2 – 25 of Kasichainula].

19. As to claim 7, Kasichainula as modified teaches responsive to the first queued component issuing a second set of method invocations, enqueueing the method

invocations of the second set into the second message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck].

20. As to claim 8, Kasichainula as modified teaches passing the interface pointer reference in queued method invocations to multiple further queued components ["Send" method 431 forwards every method name and parameters received from the intermediary object 410 to the message queue object 440; col. 6, lines 50 – 67 of Beck]; and responsive to the first queued component and the multiple queued components issuing sets of method invocations on the interface of the second queued component, enqueueing the method invocations of each such set into the second message queue [sends the name of a method and related parameters to an message queue object 440, Fig. 4; col. 6, lines 52 – col. 7, lines 23 of Beck].

21. As to claim 15, this is rejected for the same reason as claim 16 above.

22. As to claim 17, Kasichainula as modified teaches the result is a method invocation intended for the second queued component [object Y will invoke object Z' 512 when object Y's invoked method invokes the object passed in as a parameter; col. 8, lines 33 – 52 of Kasichainula].

23. As to claim 18, Kasichainula as modified teaches the result is a result of the method invoked on the first queued component [Object Z 504 returns the result 521, if any, of the invocation to object Z" 513, which returns said result 522 to object Z' 512, which returns 507 said result to object Y 503; col. 8, line 62 – col. 9, line 5 of Kasichainula].

Conclusion

24. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

U.S. Patent NO. 6,253,252 to Schofield teaches asynchronously calling and implementing objects.

25. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Li B. Zhen whose telephone number is (703) 305-3406. The examiner can normally be reached on Mon - Fri, 8:30am - 5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (703) 305-9678. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Li B. Zhen
Examiner
Art Unit 2126

lbz
February 17, 2004

[illegible]

MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100